Citation for published version:
Cliffe, O, Scotney, A & Padget, J 2010, 'Bootstrapping semantic web services with in-language annotations', 7th Extended Semantic Web Conference, Heraklion, Greece, 29/05/10 - 3/06/10.

Publication date:
2010

Link to publication

University of Bath

Alternative formats
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 06. May. 2023

Bootstrapping Semantic Web Services With In-Language Annotations*

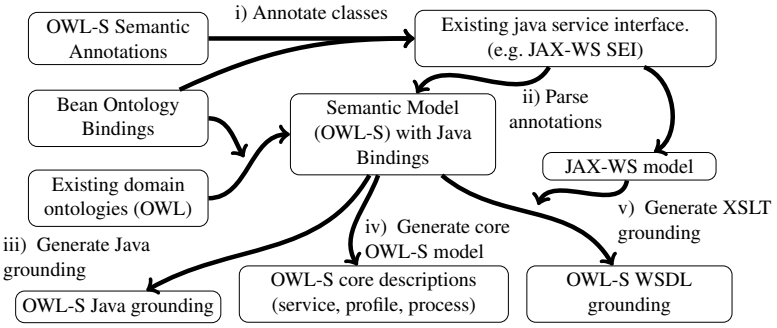
Owen Cliffe, Adan Scotney, and Julian Padget

Dept. Computer Science, University of Bath
*This work is partially supported by the IST ALIVE (FP7 215890) project
http://www.ist-alive.eu

Contribution: *We outline a mechanism and tool for the authoring and maintenance of semantic service descriptions through in-language annotations, in a manner that is analogous to that widely used for the creation of web services in high-level languages.*

Context and Problem: Semantic Web Services (SWS) offer a means to achieve meaningful automated service composition and substitution in heterogeneous service-oriented environments. Several languages (e.g. OWL-S, WSMO, SAWSDL) are used to describe (i) service semantics and (ii) bindings to the underlying service implementations. Furthermore, it is common to store semantic service descriptions separately from the service itself, using using independent description tools (e.g. Protégé, METEOR-S, WSMO-Studio). This separation poses problems when (i) creating semantic descriptions, which is potentially time-consuming and error-prone with existing languages and (ii) maintaining them, as changes must be manually tracked and corrected in the corresponding semantic description. The common approach to building web services (in high-level languages such as Java, C# Visual Basic.net etc.) is somewhat different. Rather than producing a service implementation and attaching a web-service description later, the description (in WSDL) is typically built directly from and linked closely to the program structure of the underlying implementation (or interface description): web service operation descriptions (WSDL) and underlying data formats (XML/XSD) are both derived semi-automatically from the method signatures and from the classes of data passed in and out of those methods, using a default strategy for generating corresponding web service artifacts. In the case where the desired web service or XML bindings differ from default values, in-language annotations (supported by Java 1.5, C# 2.0) on service methods and data classes are added to clarify and guide the construction process. Consequently, creating and maintaining web service descriptions is relatively simple as the transformation is driven directly by the program structure.

The Proposed Solution: We propose a mechanism and tool that apply an annotation-driven approach to the creation of *semantic* service descriptions (initially in OWL-S) for web services implemented in high-level languages (initially Java/JAX-WS), complementing the mechanism used to describe conventional web services. We contend that, in the case where a service implementation exists, then linking semantic annotations directly with the programmatic interface from which it is implemented, allows developers to create semantically-annotated services more rapidly, more easily and with fewer errors than with external descriptions.




```
@OwlsClass(defaultOntology = "http://numbers.org/Numbers.owl")
public class AddServiceSimple{
        @OwlsService(name = "AddService", label = "Addition␣Service")
        @OwlsOutParam(name = "rv", owlType = "#Integer", bindings =
            { @OwlsBinding(from = ".", to = "hasValue") })
        public int add(
           @OwlsInParam(name = "x", owlType = "#Integer",
              bindings = { @OwlsBinding(from = ".", to = "hasValue") }) int x,
           @OwlsInParam(name = "y", owlType = "#Integer",
              bindings = { @OwlsBinding(from = ".", to = "hasValue") }) int y) {
```

Fig. 1. Translation model used for building service descriptions and an example

The Workflow: Figure 1 outlines the process used for deriving semantic descriptions from JAX-WS annotated Java source (classes or interfaces) and a simple annotated service. Users first add semantic annotations (i) for service properties (profile properties, inputs, outputs, SWRL preconditions and results) and ontology bindings (bindings may be drawn from existing Java-OWL binding tools such as Jastor or JenaBean, or may be defined manually within the tool). OWLSBuilder parses these annotations and builds a semantic model of the service, including references to any existing domain ontologies (ii). This model is then used to build the required description documents including the core service description (iv). Finally, OWLSBuilder produces groundings based on the underlying mappings, currently these are direct-invocation Java groundings and WSDL groundings. In the latter case, OWLSBuilder generates XSLT transformations automatically using the parsed bindings and XML Schemas generated from JAX-WS. As with JAX-WS, sensible defaults are applied where annotations are omitted, allowing services to be created quickly with minimal annotation. Most aspects of the OWL-S model are supported (except composite processes) and the semantic binding model is flexible, and may in principle be applied to other annotation schemas such as SAWSDL in future.

Application: The tool has been used extensively within the ALIVE project, allowing developers with limited prior experience of SWS technologies to construct semantic descriptions from existing Java-based web services in a variety of domains, including intelligent communications routing, crisis management and tourism services.

The Tools: The OWLSBuilder tool is available, along with other ALIVE project software, from http://sourceforge.net/projects/ict-alive/. OWLSBuilder provides annotation support for JAX-WS services and generates OWL-S 1.2 services. A preliminary tutorial is available at https://wiki.bath.ac.uk/display/owlsbuilder/Tutorial.